Санкт-Петербургский государственный университет

Кафедра моделирования электромеханических и компьютерных систем

Тимонин Николай Олегович

Магистерская диссертация

Предварительная обработка изображений

Направление 01.04.02
«Прикладная математика и информатика»
Магистерская программа «Прикладные информационные технологии. Информационные экспертные системы»

Научный руководитель,
кандидат технических наук,
доцент
Гришкин Валерий Михайлович

Санкт-Петербург
2018

Содержание

Введение

В последнее время все в большие сферы жизни внедряются системы искусственного интеллекта (ИИ), начиная с приложений для здоровья и заканчивая персональными помощниками. В эру информационных технологий (IT) машины были помощником человека в процессе приятия решений. Сейчас же машины не только принимают решения, но и несут ответственность за их выполнение, тем самым ознаменовывая эру операционных технологий (OT).

Последние несколько лет в области компьютерного зрения явно доминируют сверточные нейронные сети (CHC) в качестве инструмента обработки. Основными предпосылками такого развития этих моделей послужило накопление достаточного количества информации для обучения, развитие алгоритмов и увеличение вычислительных мощностей.

В данной работе рассмотрена задача семантической сегментации снимков, сделанных закрепленной на автомобиле камерой в процессе движения (рис. 1). Далее снимки такого типа будем называть уличными снимками.

Рис. 1: Уличный снимок

Изображение необходимо сегментировать на основные объекты, встречающиеся во время движения автомобиля: машина, пешеход, автобус и т.д. На основании этой информации автопилот будет принимать решения об управлении автомобилем, построение траектории и маневрировании. Критически важно для данного устройства «понимание» того, что происходит

на дороге, в частности, возможность отличать пешехода от автомобиля, велосипедиста от мотоциклиста и т.д.

Актуальность данной работы обусловлена увеличением количества самоуправляемых автомобилей в последние годы. Данное изобретение позволит людям с комфортом добираться до назначенного места и проводить время в дороге с пользой, так как не надо будет управлять транспортным средством. Очевидно, что главным сдерживающим фактором прогресса в данной области является развитие систем компьютерного зрения.

Т. к. в качестве модели для сегментации используются глубокие сверточные нейронные сети (ГСНС), предметом данного исследования является изучение и построение такой архитектуры, которая позволит получить результат с высокой точностью. Одним из основных требований к процессу обучения модели является использование относительно небольшого количества вычислительных мощностей.

Постановка задачи

В данной работе были поставлены следующие задачи:

1. исследовать архитектуры сверточных нейронных сетей для семантической сегментации;
2. адаптировать выбранную архитектуру под задачу распознавания уличных снимков;
3. сформировать необходимый набор тренировочных данных;
4. обучить сеть для сегментации уличных снимков.

Обзор литературы

Первым успешным применением сверточных нейронных сетей [1] в задачах обработки изображений, а конкретно в распознавании рукописных цифр, можно, по праву, считать [2]. В этой работе автор использует нейронную сеть из семи слоев для обработки одноканальных [3] картинок размером 32×32. Возможность обработки картинок более высокого разрешения требовала больше сверточных слоев, а соответственно и вычислительных ресурсов. В дальнейшем, с развитием алгоритмов и вычислительных мощностей, СНС нашли повсеместное применение в обработке изображений: распознавание, локализация объектов, семантическая сегментация и т.д.

Семантическая сегментация изображений с использованием СНС имеет большую предысторию и одни из первых удачных попыток применений в данной области можно считать [4]. Одна из основных проблем состоит в том, что выход последнего сверточного слоя не содержит информации о точном положении объектов в виду инвариации СНС к пространственным сдвигам и поворотам. Очевидно, что данный факт очень нежелателен для задач семантической сегментации. Для преодоления данной проблемы авторы расширяют архитектуру модели путем добавления в конец полносвязного зависимого случайного поля (Conditional Random Field), которое и позволяет точно локализовывать границы объектов.

В работе [5] авторы улучшают возможности предыдущей модели путем внедрения расширенных сверточных слоев, которые позволяют расширить поле видимости фильтров. Также представлено пространственное объединение в пирамиду, которое позволяет устойчиво сегментировать объекты в разных масштабах.

С развитием сверточных сетей авторами в работе [6] был переосмыслен пространственный пирамидный пулинг. Он был расширен признаками на уровне изображения (image-level features), которые содержат глобальный контекст картинки и тем самым улучшают результаты сегментации. В данной архитектуре было принято решение отказаться от использования

полносвязного зависимого случайного поля, в виду достаточно хороших результатов и без него.

В работе [7] была представление кодер-декодер архитектура, где в качестве кодера использовалась предыдущая модель, и был добавлен простой, но достаточно эффективный декодер. Кодер-декодер архитектура позволяет эффективно кодировать контекст изображения (кодер), а затем поэтапно восстанавливать его до исходного разрешения и «очищать» результат в области границ объектов(декодер).

Расширив понятие Inception модуля [8], авторы [9] разработали поканальную раздельную свертку. Особенность данной свертки заключается в уменьшении количества необходимых операций для вычисления выхода слоя.Она может считаться Inception модулем с максимальным количеством поканальных сверток. Модель, построенная с использование сверточных слоев такого типа, получила название Xception. Данная архитектура превосходит по точности модель Inception V3 на базе данных ImageNet, хотя и имеет такое же количество параметров. Данный факт обусловлен более эффективным использованием этих самых параметров.

Глава 1. Общие понятия

1.1 Семантическая сегментация

Семантическая сегментация (попиксельная классификация) – процесс присвоения каждому пикселю изображения определенной метки (класса). В конечном результате получается множество сегментов, объединение которых покрывает все изображение (рис. 2), либо множество контуров. Пиксели, принадлежащие одному сегменту, имеют сходство по некоторой характеристике или свойству.

Другими словами, семантическая сегментация есть разделение объектов на классы по их структуре до их распознавания. Таким образом, на момент сегментации нет никакой дополнительной информации об объекте.

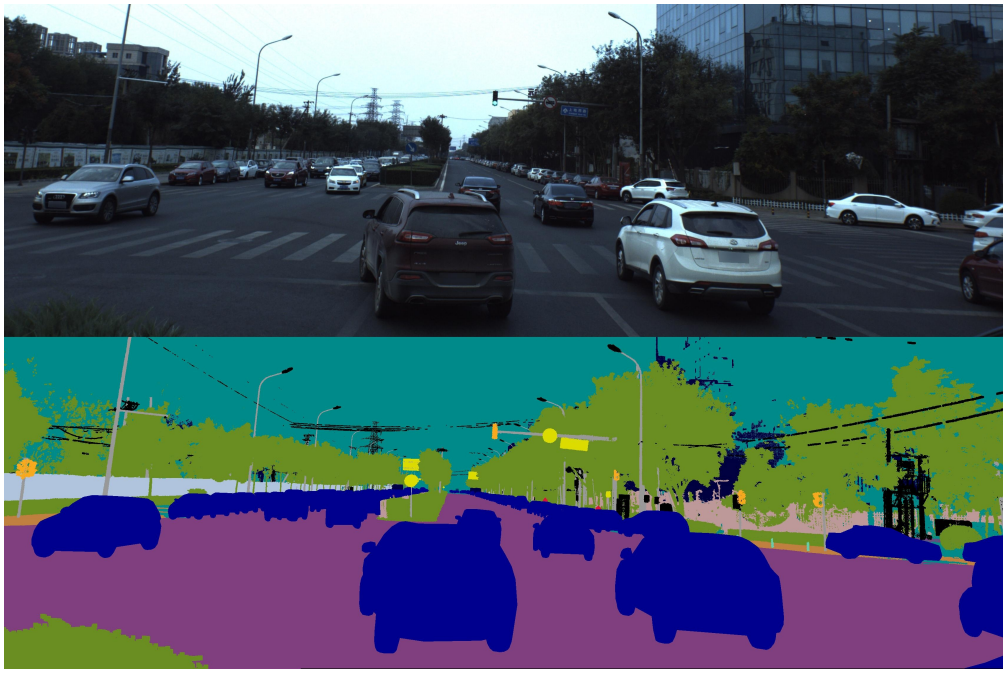


Рис. 2: Пример цветного и сегментированного изображения

В последнее время глубокие сверточные сети показывают отличные результаты в обработке изображений, поэтому было принято решение использовать их в данной работе. Инвариантность сверточных нейронных сетей играет положительную роль для таких задач, как классификация изображений, но может навредить в задачах прогнозирования высокой плотно-

сти (семантическая сегментация).

Основные проблемы семантической сегментации при использовании глубоких сверточных нейронных сетей:

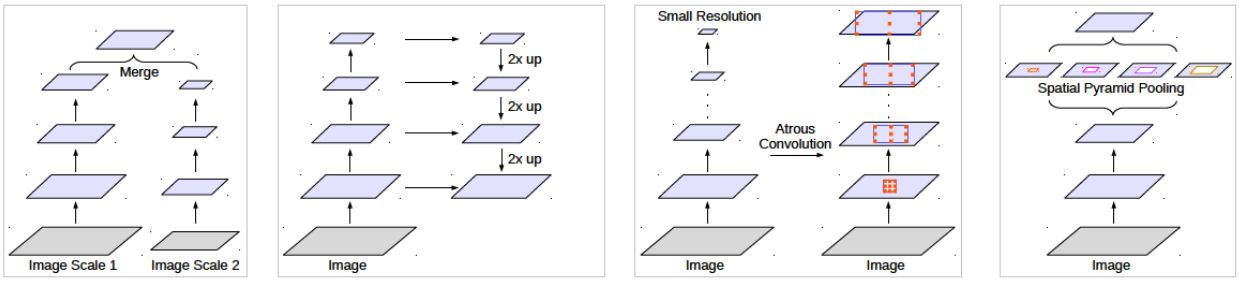
1. уменьшение пространственного разрешения;
2. существование объекта в различных масштабах;
3. уменьшение локальной точности.

Первая проблема возникает при использовании последовательных слоев уменьшения разрешения(max-pooling). Как следствие, на выходе сети карта признаков имеет существенно меньшее разрешение, нежели вход. Очевидно, что для задачи семантической сегментации это крайне нежелательно. Один из возможных способов преодоления этой проблемы - это замена всех слоев уменьшения разрешения на расширенные сверточные слои (atrous convolutions) с увеличенным коэффициентом расширения. Данная техника имеет достаточно большую предысторию и, изначально, была разработана для эффективного вычисления вейвлет преобразования. В итоге карты признаков восстанавливаются до исходного разрешения путем комбинации расширенных сверточных слоев и билинейной интерполяции в конце.

Вторая проблема обусловлена различными расстояниями от камеры до объекта в процессе съемки. Существуют четыре основных категории решения данной проблемы (рис. 3):

1. пирамида изображений (image pyramid) - через сеть пропускаются несколько вариантов одного и того же изображения в разных масштабах и далее совмещаются выходы;
2. кодер-декодер архитектура (encoder-decoder) - различные масштабы изображения преобразуются в кодере, а разрешение восстанавливается до исходного с помощью декодера;
3. надстройка дополнительных модулей на конец нейронной сети;
4. пространственное объединение в пирамиду (spatial pyramid pooling) -

входящий сигнал преобразуется фильтрами с разными степенями расширения и тем самым разными полями зрения.



(a) Image Pyramid (b) Encoder-Decoder (c) Deeper w. Atrous Convolution (d) Spatial Pyramid Pooling

Рис. 3: Различные архитектуры для обработки разных масштабов

Третья проблема обусловлена пространственной инвариантностью СНС. Особенно хорошо это наблюдается на границах сегментируемых объектов (рис. 4). Для решения данной проблемы текущая архитектура была построена по принципу кодер-декодер, где роль кодера - обработка и кодирование контекстной информации на изображении путем уменьшения разрешения, а декодера - восстановление изображения до исходного разрешения и восстановление четких границ объектов.

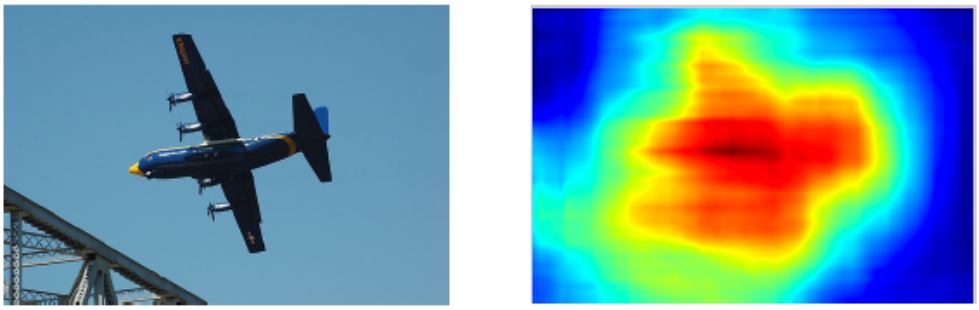


Рис. 4: Сегментация с неточными границами

1.2 Расширенные сверточные слои

Расширенные сверточные слои (atrous convolutons) - сверточные слои с расширенными фильтрами, в которых в области между ненулевыми значениями фильтра заполняются нулями. Это является обобщением обычного сверточного слоя.

В частности, для одномерного сигнала вектор выходных признаков y при одномерном входном сигнале x и фильтре w длиной K выглядит

следующим образом (рис. 5):

$$y[i] = \sum_{k=1}^{K} x[i + r * k]w[k]$$

где r - степень расширения, определяющая шаг, с которым мы отбираем значения входного сигнала. Заметим, что при $r = 1$ расширенная свертка эквивалентна обычной.

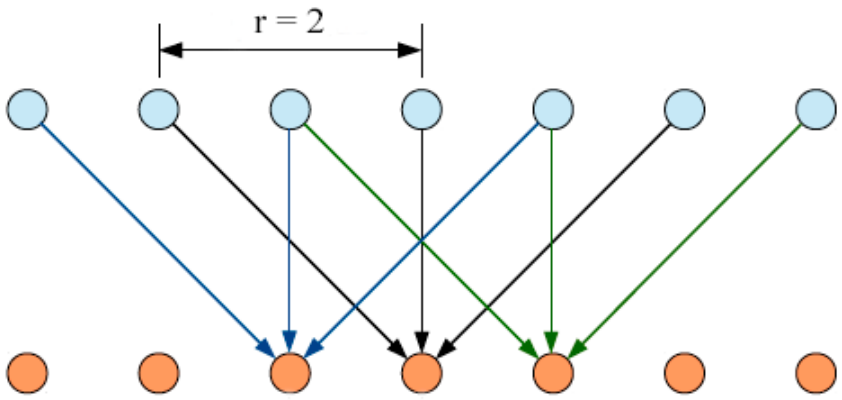


Рис. 5: Одномерная расширенная свертка

Данный тип свертки увеличивает размер ядра с $k \times k$ до

$$k_e = k + (k - 1)(r - 1)$$

где r по сути является параметром регулировки между локальной точностью (маленькое поле зрения) и ассимиляцией контекста изображения (большое поле зрения).

Пример двумерной расширенной свертки с различными коэффициентами расширения приведен на рис. 6.

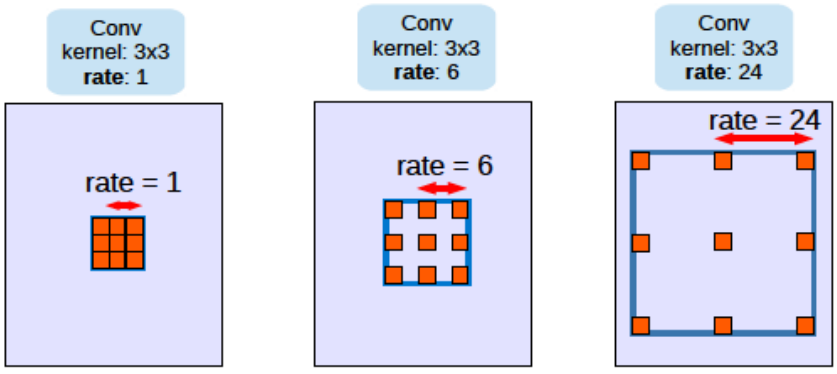


Рис. 6: Двумерная расширенная свертка

Данный тип свертки позволяет увеличивать поле зрения фильтра

для захвата информации в разных масштабах и контролировать пространственное разрешение выхода слоя. Следует отметить, что данный подход может быть альтернативой использования развертоочных (deconvolutional) слоев в задачах прогнозирования высокой плотности (семантическая сегментация). В сравнении со стандартной сверткой с большими ядрами, расширенные сверточные слои позволяют эффективно увеличить поле зрения без увеличения количества параметров.

Как правило данный тип свертки применяется в каскадном (последовательном) или пирамидном (параллельном) режиме с различными степенями расширения. Это позволяет решить проблему с нахождением объекта в разных масштабах.

1.3 Пространственное объединение в пирамиду

Несмотря на то, что сверточные сети и показывают относительную инвариантность к масштабу изображения (достигается тренировкой на наборе данных, содержащем различные масштабы одного и того же объекта), явная обработка различных масштабов может улучшить способность сети обрабатывать большие и маленькие объекты.

Один из возможных методов решения проблемы объекта в разных масштабах – создать несколько масштабов одного и того же изображения, пропустить через сеть и объединить выходы. Практически доказано, что данный подход улучшает качество сегментации. Однако, основным недостатком данного подхода является большое количество вычислений, так как нужно обработать несколько вариантов одного и того же изображения.

Для устойчивой сегментации объектов в разных масштабах модуль расширенного пространственного объединения в пирамиду (Atrous Spatial Pyramid Pooling – ASPP) преобразует входной сигнал фильтрами с разными значениями повышения дискретизации (и тем самым разными областями зрения), таким образом усваивая объекты и контекст изображения различного масштаба (рис. 7). Как правило используются четыре парал-

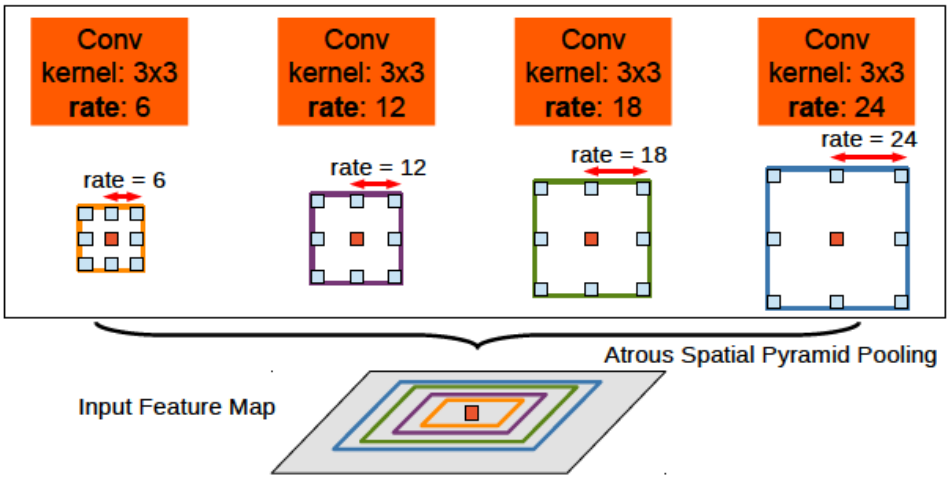


Рис. 7: Пространственное объединение в пирамиду

лельных расширенных сверточных слоя с различными степенями расширения. После это все выходы объединяются в один.

Было обнаружено, что с увеличением степени расширения, количество валидных весов фильтра (т.е. тех, которе перемножаются непосредственно на значения пикселей изображения, а не на достроенные участки) уменьшается. Так, если значение степени расширения станет близкой к разрешению карты признаков, то фильтр выродится в простой фильтр 1×1, т.к. только центральное значение фильтра будет попадать на исходные пиксели.

Для решения данной проблемы и усвоения глобального контекста изображения выделяются дополнительные признаки на уровне изображения (image level features). Процесс извлечения дополнительных признаков состоит из следующих этапов: сначала применяется глобальный усредненный пулинг (global average pooling) к последней карте признаков модели, затем 1×1 свертка с 256 фильтрами и билинейная интерполяция до требуемого пространственного разрешения.

Таким образом, в текущей архитектуре модуль расширенного пространственного объединения в пирамиду состоит из следующих частей:

1. расширенная свертка 1×1, количество фильтров 256, r = 1;
2. расширенная свертка 3×3, количество фильтров 256, r = 6;
3. расширенная свертка 3×3, количество фильтров 256, r = 12;

4. расширенная свертка 3×3, количество фильтров 256, r = 18;

5. дополнительные признаки на уровне изображения.

Выходные признаки всех пяти веток объединяются и проходят через свертку 1×1 с 256 фильтрами (рис. 8).

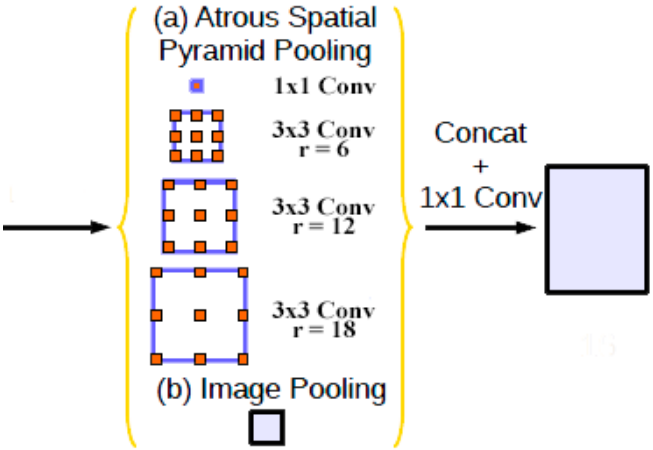


Рис. 8: Расширенное пространственное объединение в пирамиду (a) с признаками на уровне изображения (b)

1.4 Xception

В последнее время наблюдается тенденция увеличения количества слоев для улучшения качества работы сети. Однако, ограничение в производительности вычислительных машин попросту не дают обучать и использовать такие архитектуры. Одним из возможных способов решения данной проблемы является поканальная раздельная свертка (depthwise separable convolution), которая и легла в основу архитектуры Xception [9].

Классический сверточный слой обрабатывает как межканальную информацию (свёртка применяется ко всем каналам сразу), так и пространственную информацию (взаимосвязь соседних точек внутри одного канала). Предположение о том, что эти два вида информации можно обрабатывать последовательно без потери качества работы сети и легло в архитектуру Xception. Таким образом, стандартная свертка раскладывается на поканальную свертку (pointwise convolution), которая обрабатывает только межканальную корреляцию, и пространственную свертку (spatial

convolution), которая обрабатывает только пространственную корреляцию в рамках одного канала.

Два этапа поканальной раздельной свертки (рис. 9):

1. входящий тензор сворачивается ядром $1 \times 1 \times n_c$, где n_c – количество каналов входящего тензора;

2. в получившемся результате сворачивается каждый канал по отдельности сверткой $3 \times 3 \times 1$

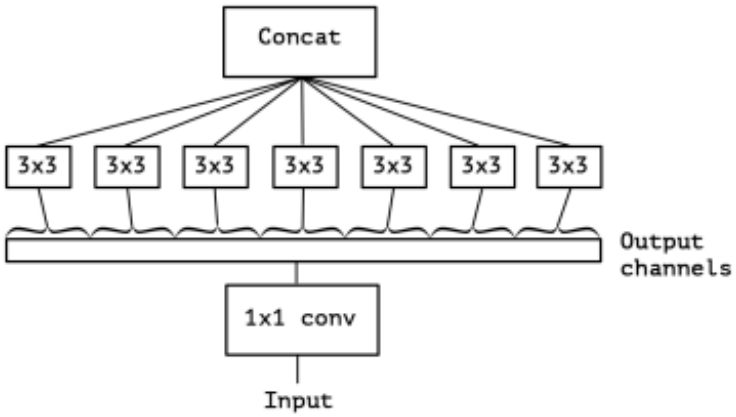


Рис. 9: Поканальная раздельная свертка

Стоит отметить, что при упоминании поканальной раздельной свертки имеется в виду, что сначала делается свертка 3×3 по каждому каналу, а затем 1×1. Практически доказано, что порядок этих операций не влияет на конечный результат.

В текущей модели в качестве кодера применяется Xception со следующими изменениями (рис. 10) :

1. все слои уменьшения пространственного разрешения (max-pool) были заменены на поканальные раздельные свертки с определенным шагом, что позволило извлекать карты признаков в произвольном разрешении;

2. дополнительно добавлены нормализация пакета тренировочных данных (batch normalization) и функция активации ReLU после каждого 3×3 поканального сверточного слоя.

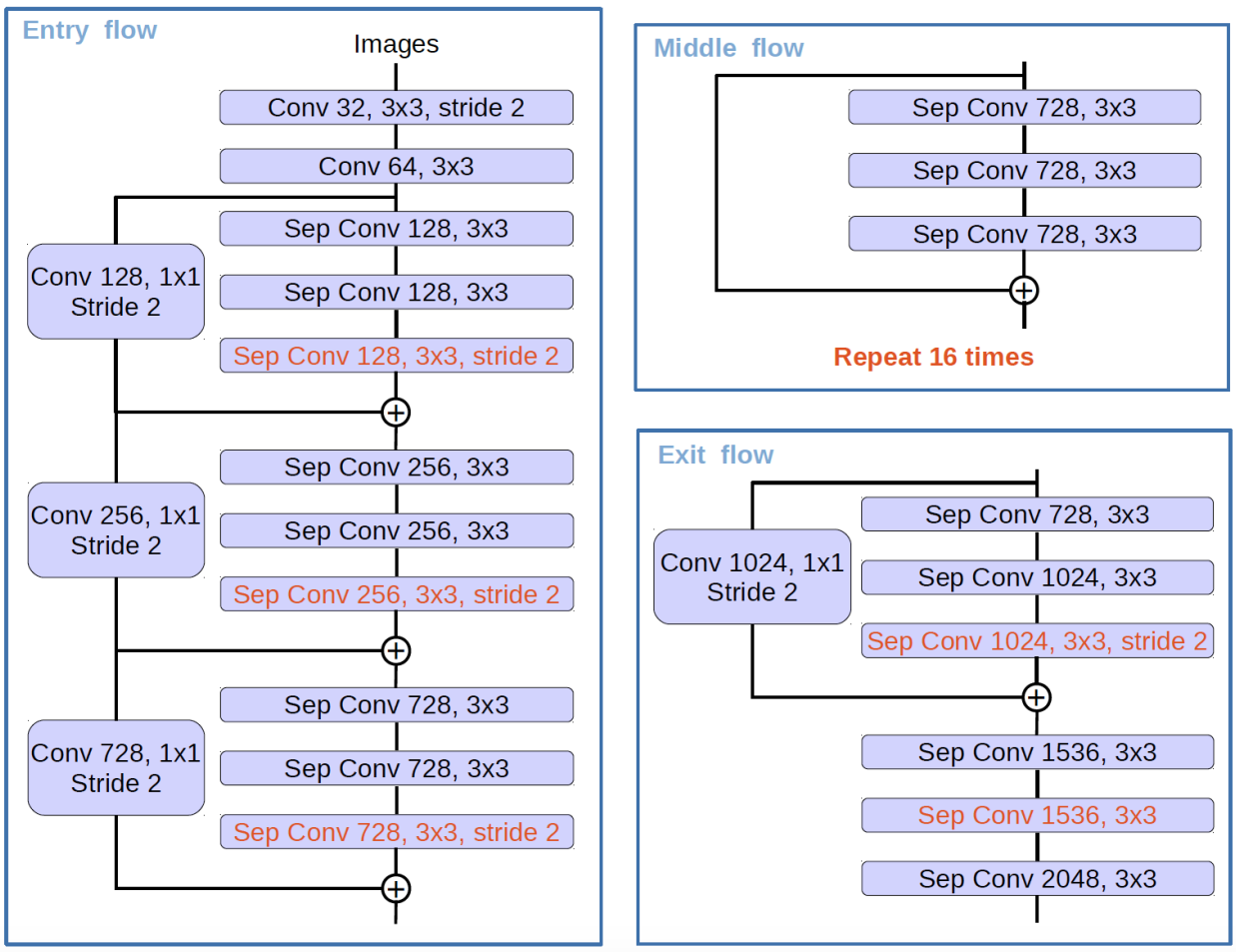


Рис. 10: Архитектура Xception

1.5 Архитектура модели DeepLabV3+

Пространственное объединение в пирамиду (spatial pyramid pooling), так же, как и кодер-декодер архитектура, используется в глубоких нейронных сетях для задач семантической сегментации. Первые способны кодировать контекстную информацию в разных масштабах путем преобразования входных признаков через фильтры или слои уменьшения разрешения с разными параметрами. Вторые способны обрабатывать четкие границы объектов путем последовательного восстановления пространственной информации. Модель DeepLabv3+ [7] (рис. 11) использует оба эти подхода, добавляя к DeepLabv3 [6] декодер для повышения точности декодирования, особенно в области границ объектов.

Используемый кодер состоит из следующих частей: адаптированная сеть Xception без полносвязных слоев и пространственного объединения

в пирамиду. На вход сети поступает изображение $513 \times 513 \times 3$. Выходное изображение меньше исходного в 16 раз. Далее данное изображение поступает на вход ASPP модуля. Напомним, что кодер может извлекать признаки с произвольным пространственным разрешением (чем больше, тем лучше). Ограничением здесь может служить только недостаток вычислительных мощностей.

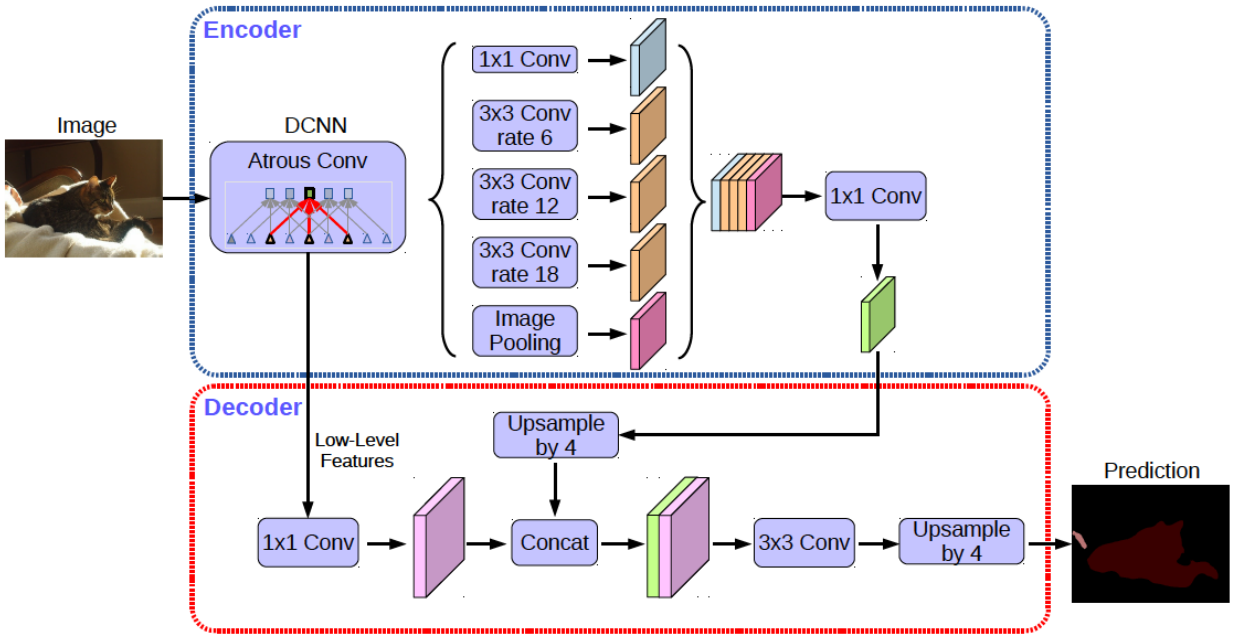


Рис. 11: Архитектура DeepLabV3+

В предыдущей архитектуре DeepLabV3 разрешение карты признаков на выходе кодера было меньше, чем на входе, в 16 раз. Данный выход может быть билинейно увеличен до исходного разрешения, что может считаться базовым декодером. Однако, такое исполнение декодера может неточно восстанавливать границы объектов. Поэтому был предложен простой и достаточно эффективный декодер. Выход кодера сначала увеличивается билинейно в 4 раза, а затем соединяется с соответствующими низкоуровневыми признаками изображения, извлекаемыми кодером, которые имеют такое же пространственное разрешение. Предварительно к этим низкоуровневым признакам применяется свертка 1×1 для уменьшения количества каналов, т.к. соответсвующие признаки обычно имеют большое количество каналов (256 или 512), что может перевешивать значимость высокоуровневых признаков (256 в текущей модели) и усложнит процесс обучения. После

соединения признаков применяется свертка 3×3 для «очистки» признаков, а затем билинейная интерполяция с множителем 4.

В модели DeepLabV3+ поканальная раздельная свертка применяется как в ASPP модуле, так и в декодере, что позволяет уменьшить количество операций для вычисления выхода слоя, и тем самым дает ускорение процесса обработки изображения.

Заметим, что увеличение выходной карты признаков кодера (т.е. на выходе кодера меньше, чем на его входе в 8 раз) дает существенное увеличение производительности модели, но требует дополнительного времени вычисления.

Глава 2. Адаптирование DeepLabV3+ под задачу сегментации уличных снимков

2.1 Тонкая настройка сети

Задача семантической сегментации требует обучения достаточно глубокой сверточной нейронной сети, однако обучение таких глубоких сетей редко производится с нуля при произвольной инициализации весов. Основные причины этого следующие: отсутствие тренировочного набора данных нужного размера и необходимость наличия достаточно мощных вычислительных ресурсов.

В данной работе предлагается использовать метод тонкой настройки сети (transfer learning). Предварительно обучается модель на большем объеме относительно схожих данных или берется уже обученная модель (задача A). Далее полученные веса используются в качестве инициализации для другой задачи (задача B), либо изменяется архитектура последних слоев сети («голова» сети). Во втором случае обучаются только новые слои, а веса предыдущих слоев («тело» сети) не обновляются в процессе обучения.

Данный подход имеет смысл если:

1. задача A и B имеют одинаковый вход;
2. имеется гораздо больше данных для задачи A нежели для задачи B.

Резюмируя все вышесказанное, к достоинствам данного подхода можно отнести:

1. возможность использования относительно малого количества элементов в дообучающей выборке;
2. возможность предварительного вычисления выходов «тела» сети для ускорения обучения;
3. достаточно быстрая возможность адаптации к конкретной задаче.

Основным недостатком данного подходя является более низкая точность полученной модели, в сравнении с моделью, обученной с нуля.

2.2 Замена выходного модуля сети

Для поставленной задачи необходима сегментация изображения на следующие классы:

1. легковой автомобиль (car);
2. мотоцикл (motorbicycle);
3. пешеход (person);
4. грузовой автомобиль (truck);
5. автобус (bus);
6. фон (background).

Исходная модель DeepLabV3+ позволяет сегментировать изображение на 21 класс: человек, кошка, собака и т.д. В данной модели был заменен выходной модуль сети (Logits), который формировал выход $1\times129\times129\times21$ на модуль с выходом $1 \times 129 \times 129 \times 6$ (рис. 12).

Полученная модель содержит около 50 млн параметров. Тренировка «с нуля» такой модели требует вычислительные ресурсы большой мощности и продолжительное количество времени. Также для тренировки такой модели и достижения высокой точности необходим достаточно большой тренировочный набор данных. При попытке тренировки модели с таким кол-вом параметров на таком маленьком наборе данных может произойти переобучение (overfitting) – модель «запомнит» все тренировочные примеры и не будет способна к обобщению информации.

Учитывая все факты, было принято решение обучать только новый модуль сети, а все оставшиеся веса «заморозить», т.е. не обновлять в процессе тренировки. Таким образом, количество обучаемых было уменьшено до:

$$1 * 1 * 256 * 6 + 6 = 1542$$

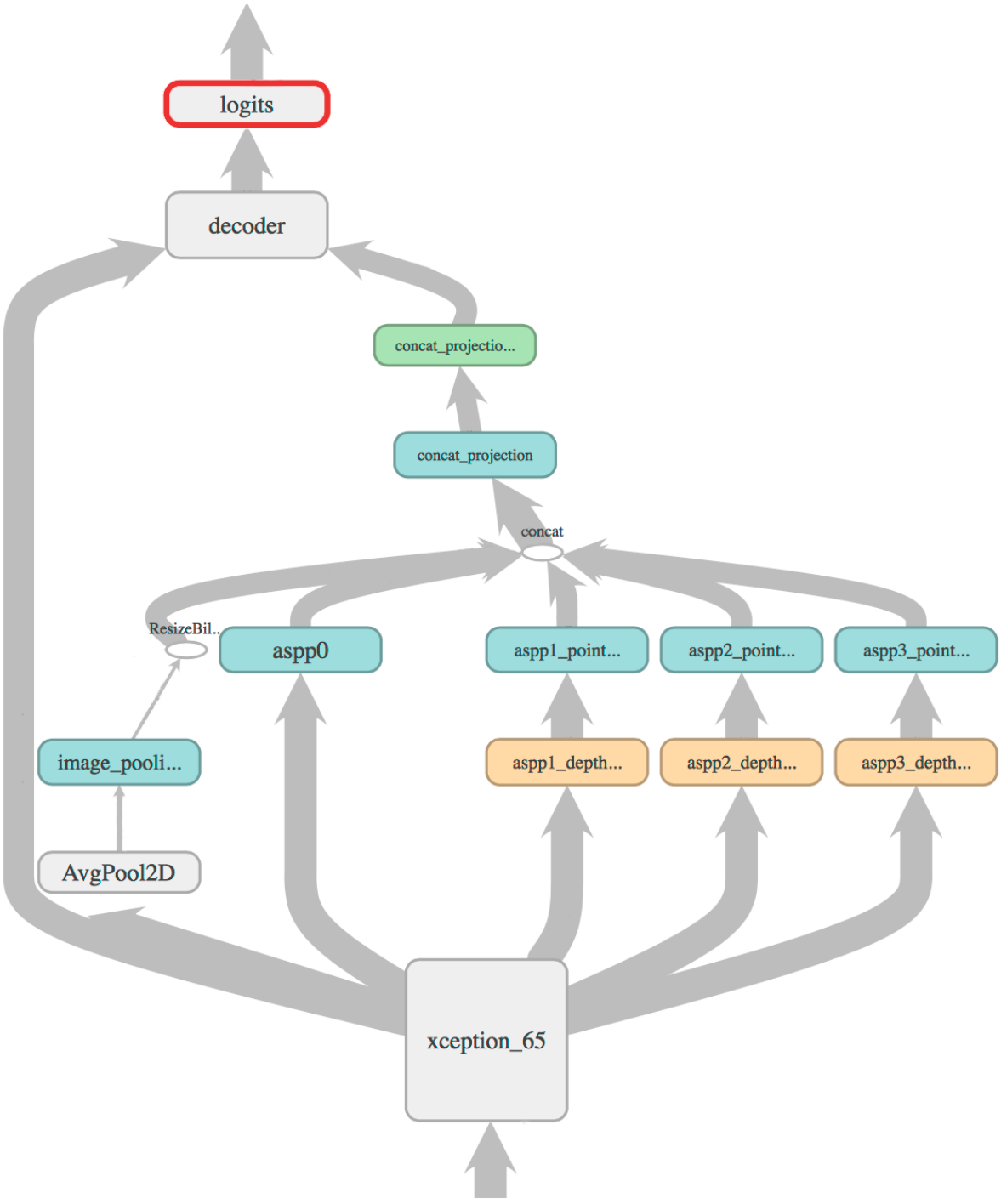


Рис. 12: Адаптирование архитектуры

Глава 3. Практическая реализация

Практическая реализация модели была выполнена в среде Python с применением библиотеки для глубокого обучения tensorflow [13].

3.1 Предварительная обработка входных данных

Набор данных состоит из 186 пар изображений. Суммарное количество объектов каждого класса приведено на рис. 13. Как и ожидалось, наиболее часто встречаются объекты класса «автомобиль», а затем «пешеход». Очевидно, что данный набор данных не сбалансирован. Это может привести к различному качеству сегментации разных классов.

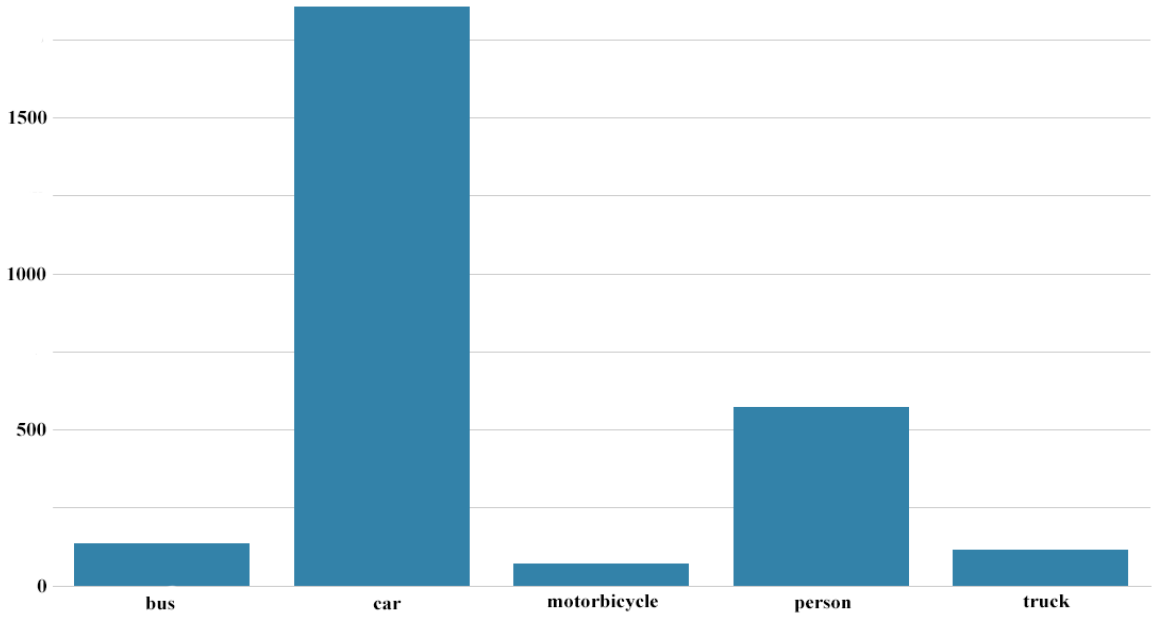


Рис. 13: Суммарное количество объектов каждого класса

Набор данных был поделен на следующие части:

1. тренировочный набор - 112 пар (60 %). Используется непосредственно для тренировки модели.
2. валидационный набор - 37 пар (20 %). Применяется для подбора параметров модели.
3. тестовый набор - 37 пар (20 %). Для финальной оценки качества модели.

Каждая пара изображений (рис. 14) состоит из:

1. изображения, снятого камерой с автомобиля в формате JPG с разрешением $3384 \times 2710 \times 3$. Далее такие изображения будем называть цветными.
2. сегментированного изображения в формате PNG с разрешением $3384 \times 2710 \times 1$. Класс пикселя закодирован непосредственно в первые две цифры его значения: автомобиль - 33, мотоцикл - 34, грузовик - 38 и т.д.

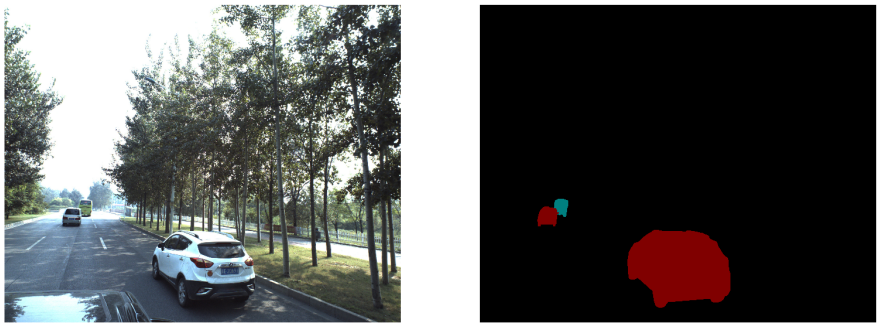


Рис. 14: Одна пара изображений

На первом этапе обработки обрабатываются только сегментированные изображения. Каждому пикселю изображения присваивается новый код класса: фон - 0, автомобиль - 1, мотоцикл - 2 и т.д. Это позволит нам в дальнейшем отобразить данные значения в вектора (one-hot encoding).

Далее и цветное изображение и сегментированное расширяются до размеров $3384 \times 2712 \times n_c$, а затем делятся на девять одинаковых участков с размерами $1128 \times 904 \times n_c$, каждый из которых уменьшается до размеров $513 \times 513 \times n_c$, где n_c - количество каналов (рис. 15). После этого удаляются те пары извлеченных изображений, на которых содержится только фон. Данная операция обусловлена следующими факторами:

1. размер входа сети $513 \times 513 \times 3$ и уменьшение исходного изображения до такого разрешения «съест» небольшие детали;

2. более сбалансированный тренировочный набор.

Таким образом, после обработки тренировочный набор содержал 352 пары изображений, валидационный - 126, тестовый - 99.

Рис. 15: Разделенное изображение

3.2 Вычисление выходов декодера DeepLabV3+

В данном исследовании было принято решение предварительно вычислить выходы декодера модели для каждого тренировочного изображения. Это позволит существенно ускорить процесс обучения, поскольку большая часть вычислений происходит именно в «теле» (рис. 16) сети. Причем каждое изображение пропускается через сеть много раз. Таким образом, на вход сети последовательно поступают цветные изображения разме-

ром $513 \times 513 \times 3$ и далее выходы декодера $129 \times 129 \times 256$ сохраняются на диске.

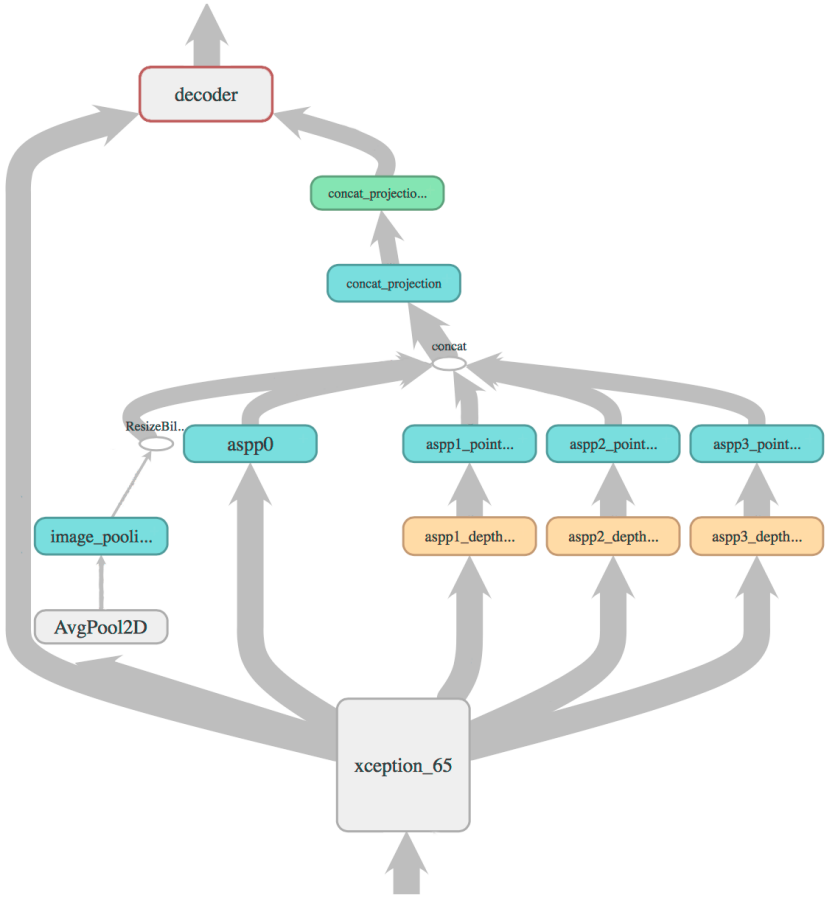


Рис. 16: Основная часть модели

3.3 Обучение выходного модуля

Новый модуль Logits состоит из одного сверточного слоя с ядром свертки $1 \times 1 \times 256$ (рис. 17). Всего таких ядер шесть, по одному на каждый класс. Также в данном модуле используется L2-регуляризация [10] для уменьшения степени переобучения модели.

Процесс обучения происходит следующим образом. Из декодера на вход модуля поступает сигнал размерностью $129 \times 129 \times 256$. Выход модуля обозначим a, размерность которого $129 \times 129 \times 6$.

$$a = \{a_{i,j,k}\}, i = 1, ..129; j = 1, ..., 129; k = 1, ..., 6.$$

Затем к каждому вектору $a_{i,j,k}, i = 1, ..129; j = 1, ..., 129$ размерности $1 \times 1 \times 6$ применяется softmax функция. Данная функция преобразует

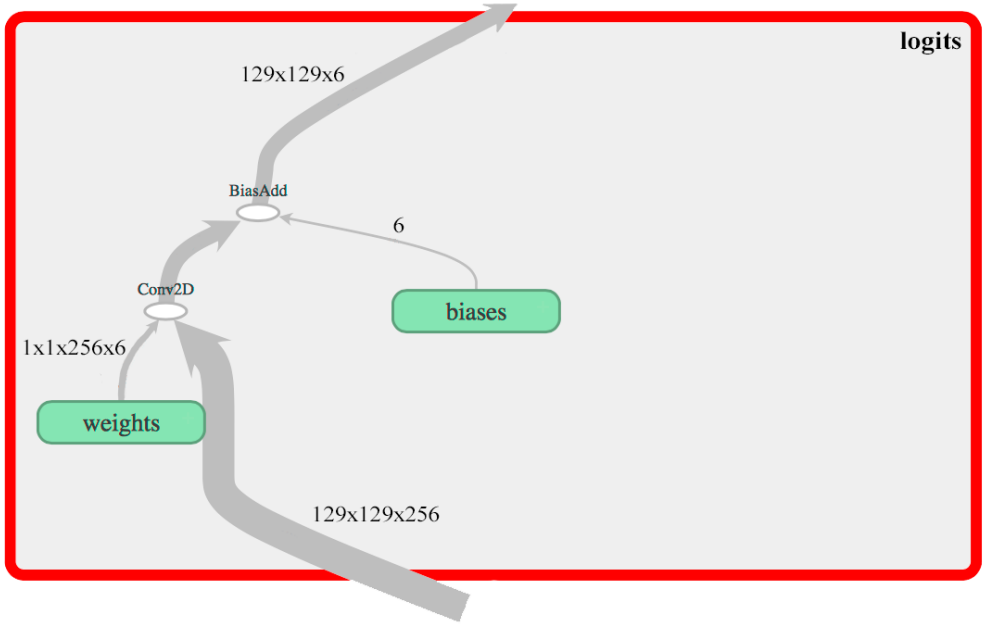


Рис. 17: Logits модуль

вектор в вектор такой же размерности, где каждая координата лежит в вещественном интервале $[0, 1]$ и сумма координат равна единице:

$$R^{n_c} \implies [0,1]^{n_c},$$

$$\hat{y}_{i,j,k} = \frac{e^{a_{i,j,k}}}{\sum\limits_{l=1}^{n_c} e^{a_{i,j,l}}}, k = 1, .., n_c,$$

$$\sum_{k=1}^{n_c} \hat{y}_k = 1.$$

В данном случае $n_c = 6$, т.к. классов шесть. Таким образом, значение каждого элемента такого вектора является вероятностью реализации случайной величины. Индекс элемента в векторе и будет классом пикселя. В итоге преобразованный выход модуля:

$$\hat{y} = \{\hat{y}_{i,j,k}\}, i = 1, ..129; j = 1, ..., 129; k = 1, ..., 6.$$

Соответствующее сегментированное изображение уменьшается до размеров $129 \times 129 \times 1$ и расширяется до $129 \times 129 \times 6$, чтобы соответствовать выходу модуля Logits. Обозначим его следующим образом:

$$y = \{y_{i,j,k}\}, i = 1, ..129; j = 1, ..., 129; k = 1, ..., 6.$$

В качестве функции потерь используется функция кросс-энтропии [11]. Значение этой функции считается следующим образом:

$$L = \sum_{i=1}^{n_h} \sum_{j=1}^{n_w} l_{i,j}, n_h = 129, n_w = 129, \text{ где}$$

$$l_{i,j} = -\sum_{k=1}^{n_c} y_{i,j,k} \ln \hat{y}_{i,j,k}, i = 1, .., 129, j = 1, ..., 129, n_c = 6.$$

В качестве оптимизационного алгоритма для уменьшения функции потерь был выбран ADAM [12]. Коэффициент скорости обучения (learning rate) равен 10^{-3}. Коэффициент регуляризации – 10^{-4}

В качестве метрики на валидационном множестве была выбрана метрика пересечение над объединением (Intersection over Union – IoU). Она вычисляется как отношение площади пересечения реального объекта и предсказанного к их объединению (рис 18). В данном исследовании использовалось среднее значение данной метрики по всем классам (mIoU). Заметим, что данная метрика не может быть использована для оптимизации, так как не является дифференцируемой.

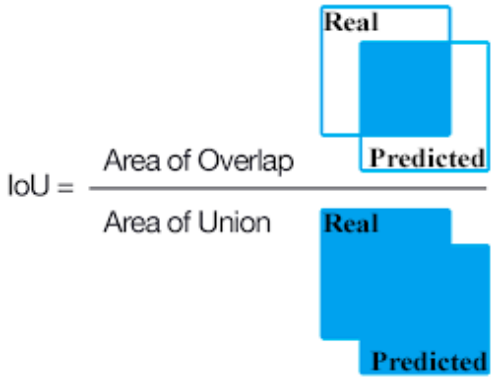


Рис. 18: Intersection over Union

График зависимости ошибки на тренировочном множестве от количества итераций представлен ниже. Заметим, что после определенного количества итераций (около 1500) достигается «плато» как в уменьшении ошибки, так и в увеличении точности. Данный факт свидетельствует о том, что все настраиваемые параметры модели выбраны верно и не происходит переобучение.

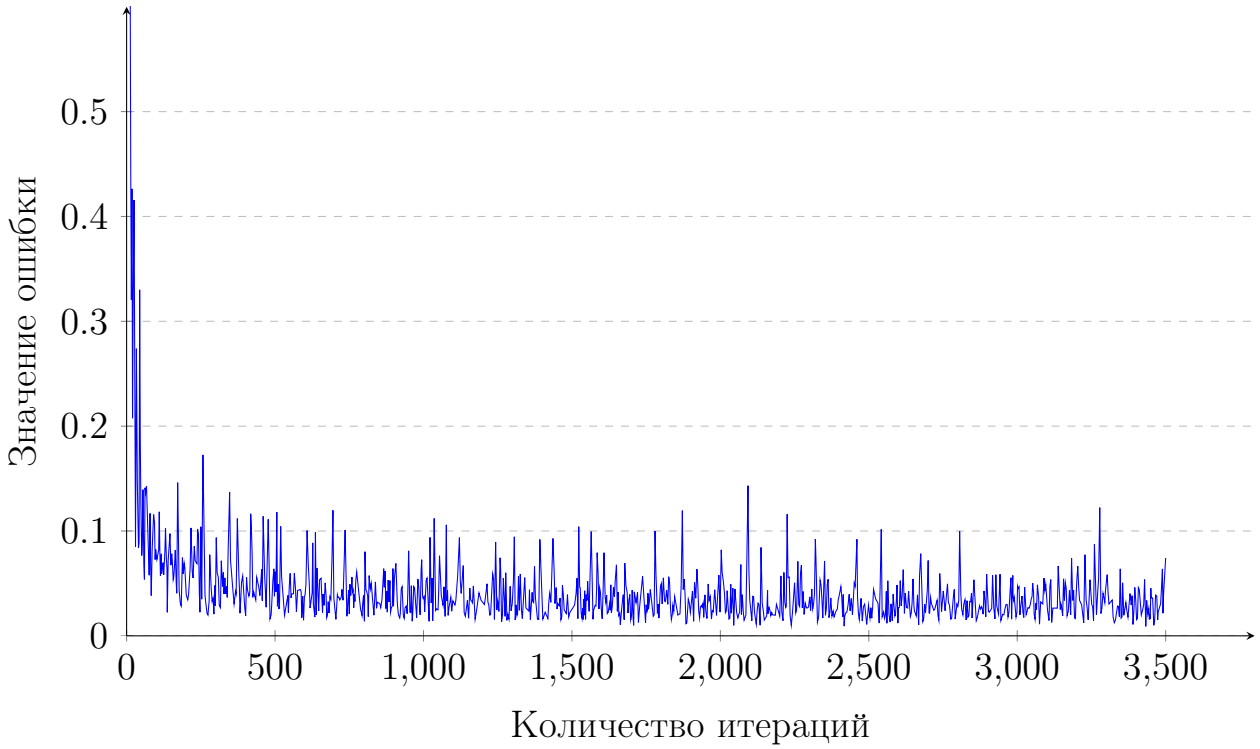
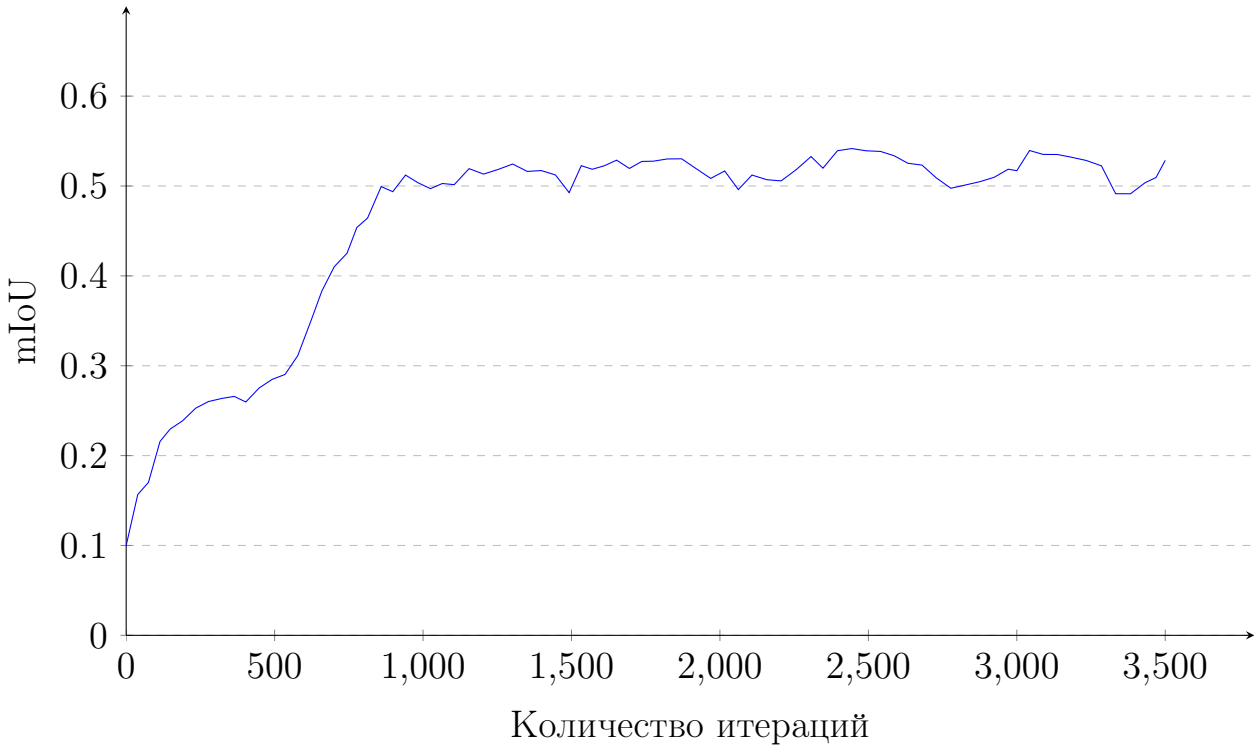


График значения метрики IoU на валидационном множестве от количества итераций представлен ниже:



После обучения на тестовом множестве значение метррики mIoU составляло 0.45 или 45%.

3.4 Визуализация результата работы программы

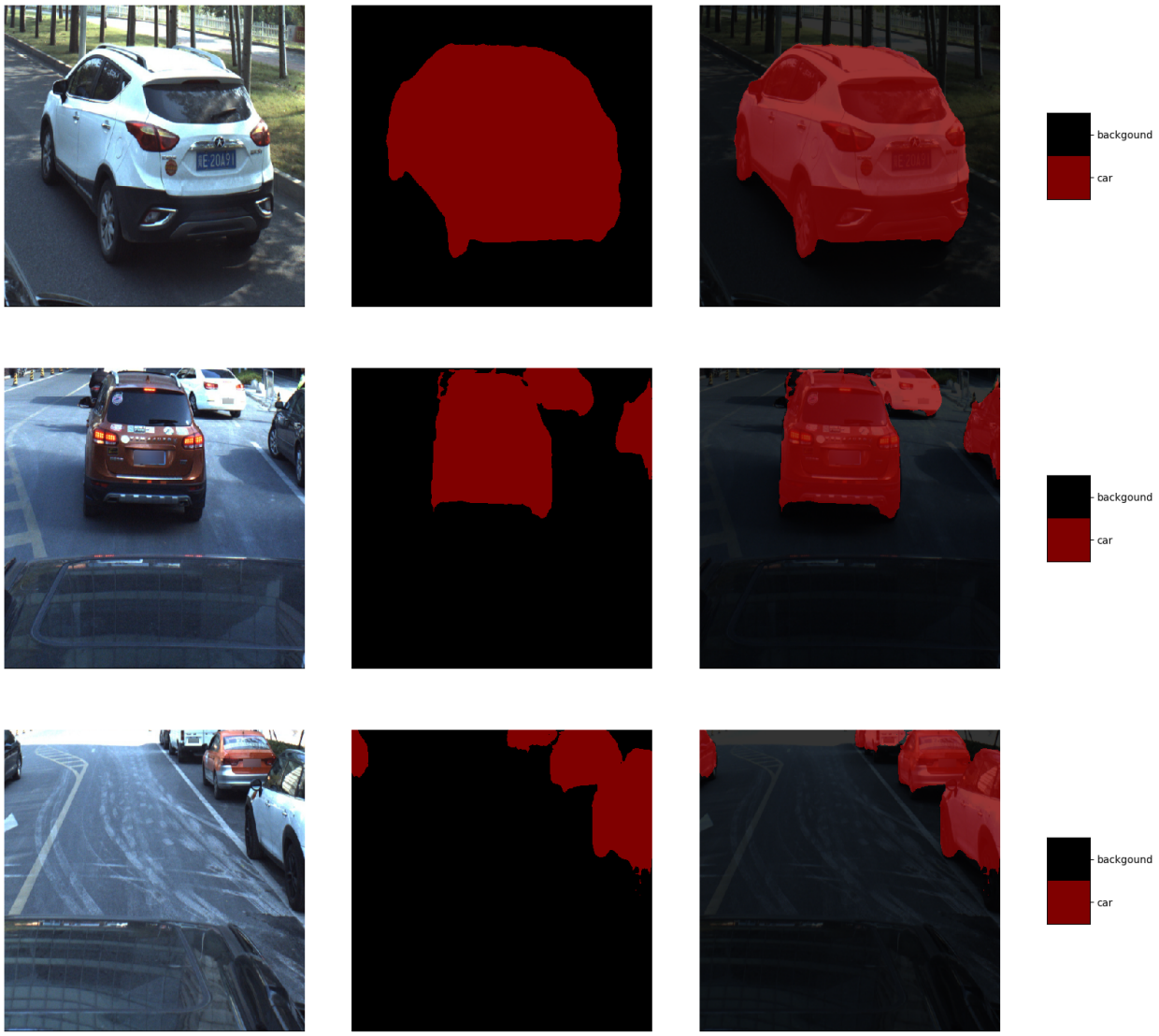


Рис. 19: Сегментация легковых автомобилей

Визуализация работы программы проводилась на снимках, принадлежащих тестовому множеству. Как видно, на рис. 19 модель отлично справляется с сегментированием легковых автомобилей. Нар рис. 20 представлены примеры сегментации автомобилей и автобусов. В этом случае также можно отметить высокое качество обработки. На рис 21 представлены примеры сегментации автомобилей и пешеходов. Можно отметить среднее качество сегментации пешеходов, что обусловлено двумя причинами: несбалансированный набор тренировочных данных и относительно маленькое количество пикселей на изображении, принадлежащих классу «пешеход».

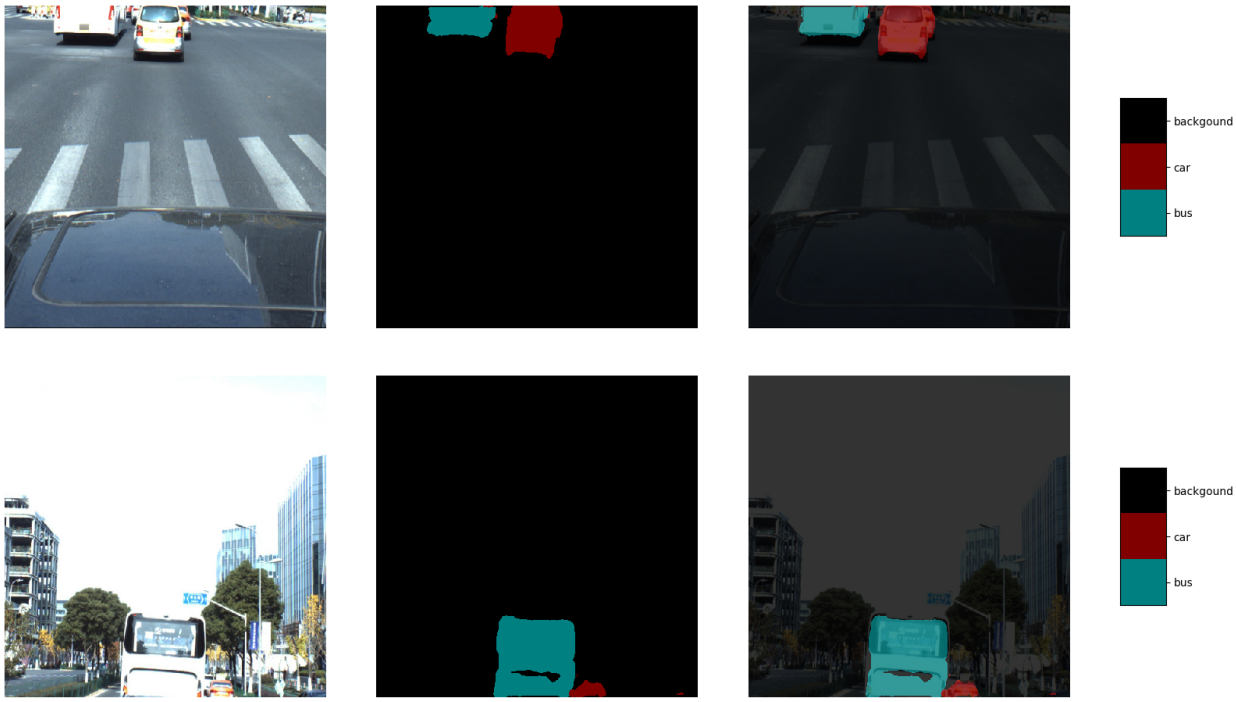


Рис. 20: Сегментация автомобилей и автобусов

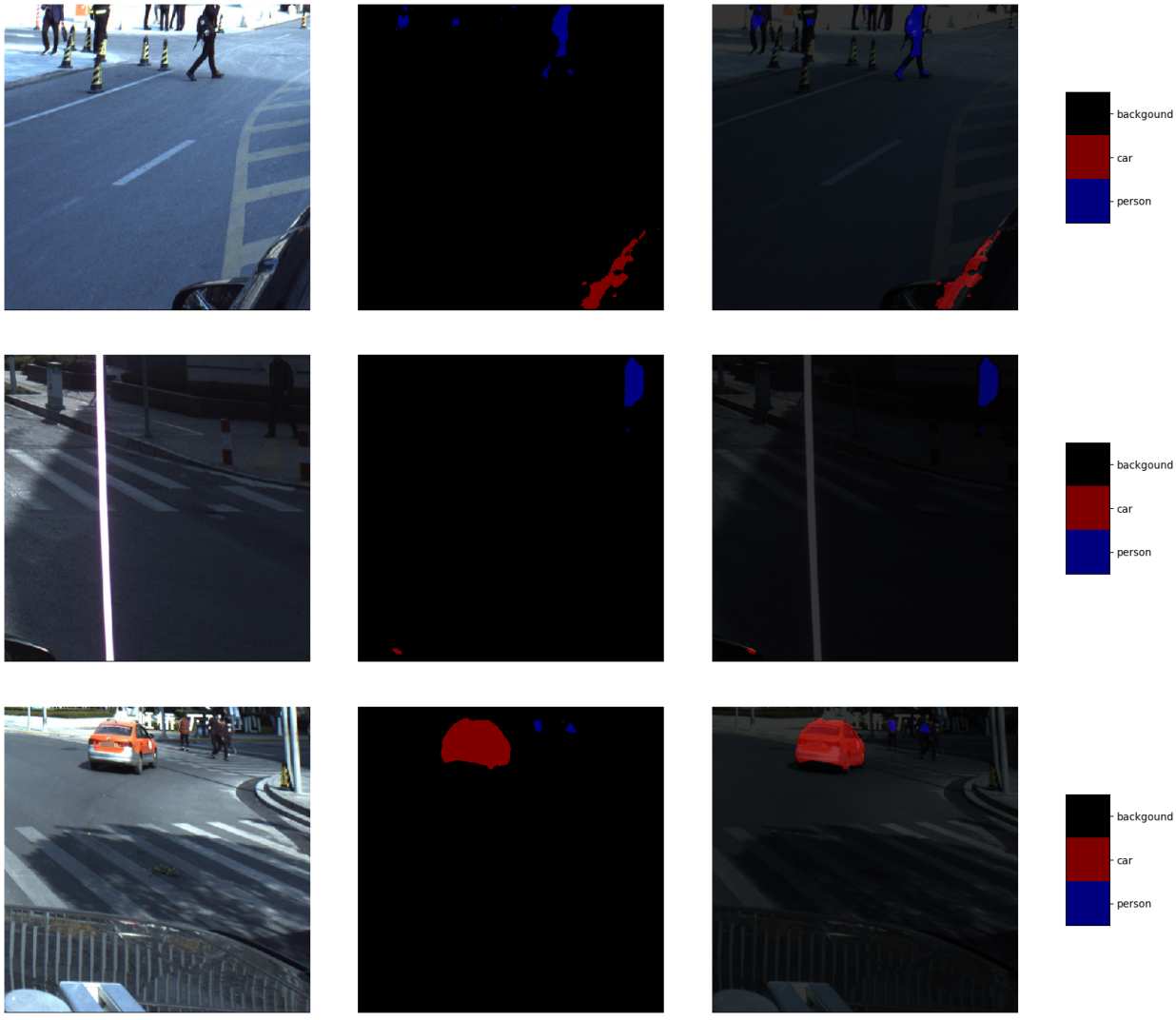


Рис. 21: Сегментация автомобилей и пешеходов

Вывод

Удаление картинок из тренировочного набора, которые содержат только фон положительно сказывается на процессе обучения. Данный факт обусловлен более сбалансированным тренировочным набором и отсутствием изображений, на которых сеть ничему не обучается.

Данное исследование показывает, что использование метода точной настройки уже построенной архитектуры под конкретную задачу существенно уменьшает требования к размеру набора тренировочных данных. Такой подход также рекомендуется использовать в условиях ограниченного количества вычислительных ресурсов.

Предварительный расчет выходных данных декодера позволяет сильно ускорить процесс обучения, однако требует дополнительного дискового пространства для хранения промежуточных результатов. Заметим, что количество требуемого пространства не столь велико в сравнении с существенным ускорением обучения. Стоит отметить, что существует возможность тренировки модели непрерывной цепью без сохранения промежуточного результата, т.е. на вход будет подаваться цветное изображение, а выход сети будет сравниваться с сегментированным изображением. Однако, в данной работе было принято решения отказаться от него и тренировать модель в два этапа из-за высокой стоимости вычисления выхода декодера.

Достигнутое высокое значение метрики пересечения над объединением на тестовом множестве. Как и предполагалось, модель лучше справляется с сегментацией объектов класса «автомобиль» и «автобус», а хуже всего с «пешеход» , «мотоцикл» и «грузовой автомобиль». Данный факт связан с несбалансированностью тренировочного набора. Для улучшения показателей можно использовать тренировочный набор с большим количеством пешеходов, мотоциклистов и грузовых автомобилей или расширить текущий.

Также в данном исследовании стояла задача организации тренировочного процесса на данных, которые не умещаются полностью в опера-

тивной памяти. Один из возможных способов решения данной проблемы, который и был реализован в данной работе – организация отдельной очереди подгрузки очередного набора тренировочных примеров, необходимых для следующей итерации. Этот подход позволяет избежать простаивания ресурсов при считывании нового набора примеров с диска.

Заключение

Анализ задачи сегментации изображений с помощью СНС показал, что ключевую роль в достижении достаточно точного результата здесь играют архитектура выбранной модели, тренировочный набор данных и вычислительные мощности. После исследования различных архитектур была выбрана модель DeepLabV3+, как наиболее новая модель, показывающая лучшие результаты на тесте PASCAL VOC [14].

Выбранная модель была адаптирована под задачу распознавания уличных снимков путем замены Logits модуля и его тренировки. Основанием для данного решения послужили небольшой набор тренировочных данных и отсутствие больших вычислительных мощностей.

Далее был сформирован и обработан необходимый тренировочный набор данных путем разделения каждой картинки на девять одинаковых частей и удаления из них тех, которые содержат только фон.

Было написано программное обеспечение с использованием библиотеки для машинного обучения tensorflow для обработки тренировочных данных, создания модели и последующего ее обучения. В процессе обучения была достигнута выскокая точность на тестовом множестве.

Данное исследование может быть расширено следующим образом:

1. обучение не всего модуля Logits с нуля, а только той его части, которая не пересекается с модулем исходной модели;
2. обучение не только модуля Logits, но и нескольких предыдущих слов;
3. обучение на расширенном и более сбалансированном тренировочном наборе данных.

Список литературы

[1] Хайкин С. Нейронные сети. Полный курс. Вильямс, 2006. 1105 с.

[2] LeCun Y., Bottou L., Bengio Y.,Haffner P. Gradient-based learning applied to document recognition // Proceedings of the IEEE, 1998.

[3] Гонзалес Р. С., Вудс Р. Е. Цифровая обработка изображений. Москва, Техносфера, 2012. 1070 с.

[4] Chen L., Papandreou G., Kokkinos I., Murphy K., Yuille A. L. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs // Proc. of ICLR, 2015.

[5] Chen L., Papandreou G., Kokkinos I., Murphy K., Yuille A. L. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs // TPAMI, 2017.

[6] Chen L., Papandreou G., Schroff F., Adam H. Rethinking Atrous Convolution for Semantic Image Segmentation // arXiv:1706.05587, 2017.

[7] Chen L., Yukun Z., Papandreou G., Schroff F., Adam H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation // arXiv: 1802.02611, 2018.

[8] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the Inception Architecture for Computer Vision // arXiv:1512.00567, 2015.

[9] Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions // Proc. of CVPR, 2017.

[10] Cortes. C, Mohri M., Rostamizadeh A. L2 Regularization for Learning Kernels // arXiv:1205.2653, 2012.

[11] Liu W., Wen Y., Yu Z., Yang M. Large-Margin Softmax Loss for Convolutional Neural Networks // arXiv:1612.02295, 2017.

[12] Kingma D., Ba J. Adam: A Method for Stochastic Optimization // arXiv:1412.6980, 2014.

[13] Machine learning framework. https://www.tensorflow.org

[14] PASCAL VOC segmentation benchmark. http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html